Mandatory exercise for INF 144
Deadline: Sunday March 22, 2015

The exercise is to make programs (in your favorite programming language, e.g., Java, Python, or C/C++) that analyze and compress the folktale "Askeladden som stjal sølvendene til trollet" which can be found at

`http://folk.uib.no/st03333/INF144_2015/Oblig/folktale.html`

Please upload the source code, instructions how to compile and run it, and a document describing what you have done (including the answers to the questions below) to MiSide by the deadline, which is Sunday March 22, 2015.

Generate random text

You should make a program that generates random Norwegian text based on a Markov model as follows:

- Zeroth order approximation: The 30 letters (including a single space) are chosen independently of each other. The source statistics should be found by analyzing the folktale mentioned above. Describe the model in the report.
- First order approximation: Make a Markov model that have a state for each of the 30 possible letters (including a single space). The transition probabilities of the model should be found by analyzing the folktale mentioned above. Describe the model in the report.
- Second order approximation: Make a Markov model that have a state for each pair of the 30 possible letters (including a single space). The transition probabilities of the model should be found by analyzing the folktale mentioned above. Describe the model in the report.
- Third order approximation: Make a Markov model that have a state for each triple of the 30 possible letters (including a single space). The transition probabilities of the model should be found by analyzing the folktale mentioned above. Describe the model in the report.

For each of the approximations above describe what you observe. In particular:

1. Do you recognize some real words in the random text?
2. Are the Markovian information sources unifilar, and if so explain why they are unifilar and compute their entropy.

Lempel-Ziv-Welch and Huffman compression

- Implement the Lempel-Ziv-Welch (LZW) compression algorithm as described in the lecture nodes. In particular, you should implement both the compression and decompression operations and it should work on a text file containing the 29 Norwegian letters and a space. Test the implementation on the folktale mentioned above and report the compression rate.
- Augment the previous implementation by Huffman encoding, i.e., the program should apply Huffman encoding to the LZW output. The source statistics can be found from the LZW output. Test the implementation on the folktale mentioned above and report the overall compression rate. Are you able to increase the compression rate?
- Repeat the exercise on 100 randomly generated texts (of the same length as the folktale mentioned above) for each of the Markov model approximations mentioned above. Report the average compression rate with and without Huffman encoding on the LZW output.